

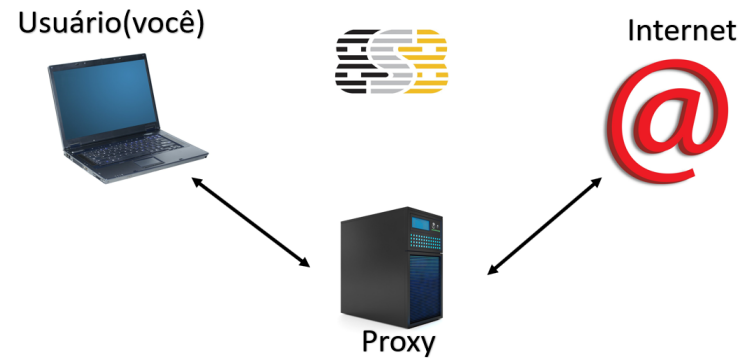
HORA DO BACK-END

Aula -1

-FERNANDO LUCAS (MASSAN)

INICIANDO O SERVIDOR.

Em informática, um servidor é um software ou computador, com sistema de computação centralizada que fornece serviços a uma rede de computadores, chamada de cliente. Esses serviços podem ser de naturezas distintas, como por exemplo, arquivos e correio eletrônico.



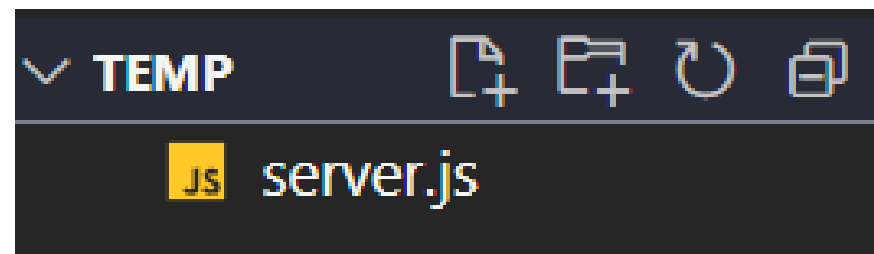
CRIANDO NOVO PROJETO



Vamos criar um novo projeto para iniciarmos nossos estudos de back-end.



Dentro dessa pasta vamos criar nosso primeiro arquivo.



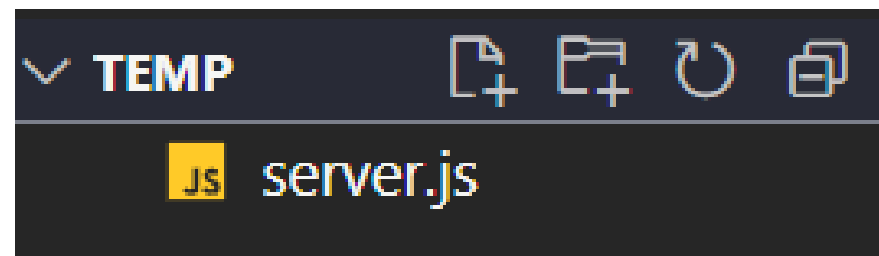
CRIANDO NOVO PROJETO



Vamos criar um novo projeto para iniciarmos nossos estudos de back-end.



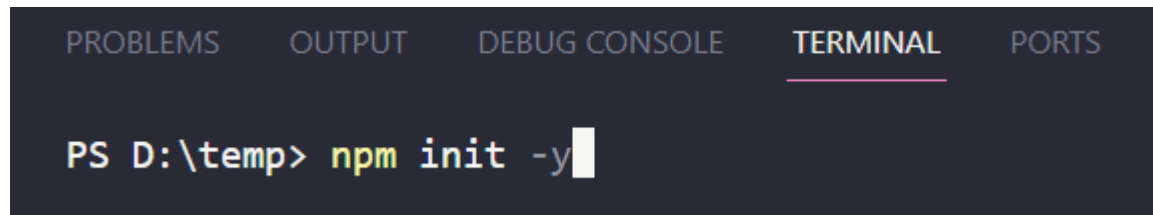
Dentro dessa pasta vamos criar nosso primeiro arquivo.



INSTALANDO DEPENDÊNCIAS

npm é o Gerenciador de Pacotes do Node (Node Package Manager) que vem junto com ele e que é muito útil no desenvolvimento Node. Por anos, o Node tem sido amplamente usado por desenvolvedores JavaScript para compartilhar ferramentas, instalar vários módulos e gerenciar suas dependências.

Abra o terminal e digite o seguinte comando!



A screenshot of a terminal window from a code editor. The terminal has a dark background. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. Below the tabs, the prompt 'PS D:\temp>' is followed by the command 'npm init -y' in a light blue font, with a white cursor at the end of the command.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\temp> npm init -y
```

ENTENDENDO O package.json

JSON é uma notação que permite estruturar dados em formato texto para serem utilizados em diferentes tipos de sistemas. Trata-se de um formato simples e leve, que oferece uma série de benefícios, como maior velocidade no tráfego em rede e mais agilidade no processamento.

 package.json
 server.js

```
package.json > ...  
1  {  
2    "name": "temp",  
3    "version": "1.0.0",  
4    "description": "",  
5    "main": "server.js",  
6    "scripts": {  
7      "test": "echo \"Error: no test specified\" && exit 1",  
8      "start": "node server.js"  
9    },  
10   "keywords": [],  
11   "author": "",  
12   "license": "ISC"  
13 }  
14
```

VAMOS INSTALAR O TEMPLATE EXPRESS

Execute o comando `npm install express` note que agora nos nosso arquivo `package.json`

Temos uma nova dependência o

```
"dependencies": {  
  "express": "^4.17.1"  
}
```

. Repare que também temos uma pasta nova: `> node_modules`. Dentro dessa pasta temos várias dependências que o express precisa para funcionar. Você pode interpretar cada dependência dessa como um programa.

ATENÇÃO



A partir de agora vamos utilizar o JavaScript puro, se você sentir muita dificuldade você deve voltar até o slide "iniciando Javascript" para rever os conteúdos iniciais.

iniciando javascript

Conhecendo o Javascript
Comentários, Strings e Numbers,
Fazendo cálculos com JavaScript,
operadores relacionais e comparativos
Condicionais,
objetos e
Arays.



PowerShell



que a força esteja com você.
Prof. fernando Lucas

NO NOSSO ARQUIVO server.js

Vamos iniciar os comandos em JavaScript

```
const express = require('express')  
const server = express()  
  
server.listen(5000, function() {  
  console.log("server is running")  
})
```

```
npm start
```

CRIANDO ROTAS

Se você acessar servidor no seu navegador "localhost:5000" você perceberá que ele pedirá uma rota. **Cannot GET /**

A rota inicial de qualquer servidor é a "/", então mesmo que a gente não coloque ela no código ela sempre será o primeiro lugar que o código irá procurar.

CRIANDO ROTAS

Já que ele não está encontrando a rota para "/", precisamos indicar pra ele onde encontrar esse lugar.

```
server.get("/", function(req,res){})
```

Vamos utilizar uma função com parâmetros "req" e "res".

Vamos entender essa linha de código.

CRIANDO ROTAS

`server.get` Servidor pegue algo.

`server.get("/",` Servidor pegue a barra

`server.get("/", function` servidor pegue essa barra e execute essa função.

`(req, res)` req = requisição (servidor escuta algo) | res= resposta (servidor responde algo).

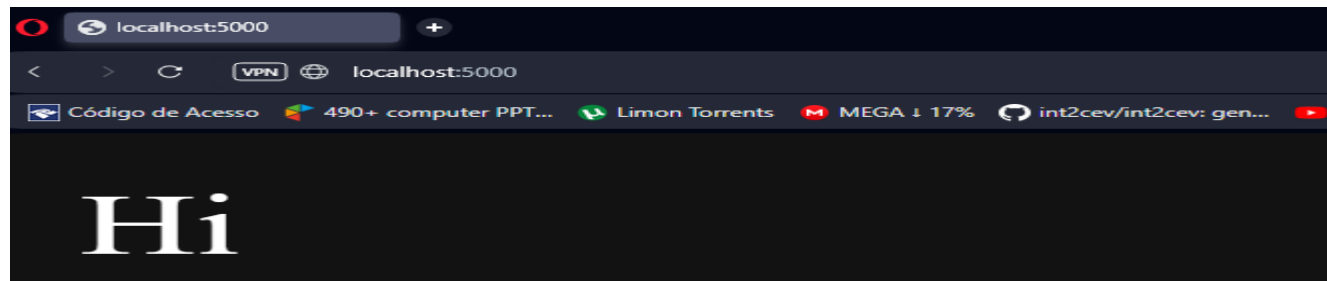
`server.get("/", function(req, res)`

Servidor pegue essa barra e execute a seguinte função, se o usuário enviar algo guarde no parâmetro "req",
Se você for responder algo guarde a resposta no parâmetros "res".

CRIANDO ROTAS

Vamos agora ordenar que o servidor responda a rota "/".

```
server.get("/", function(req,res){  
  ⚡ return res.send("Hi");  
})
```




INSTALANDO DEPENDÊNCIAS

Vamos precisar reiniciar o servidor várias vezes agora.

Para não precisarmos reiniciar toda hora manualmente vamos instalar uma extensão que vai fazer isso de forma automática.

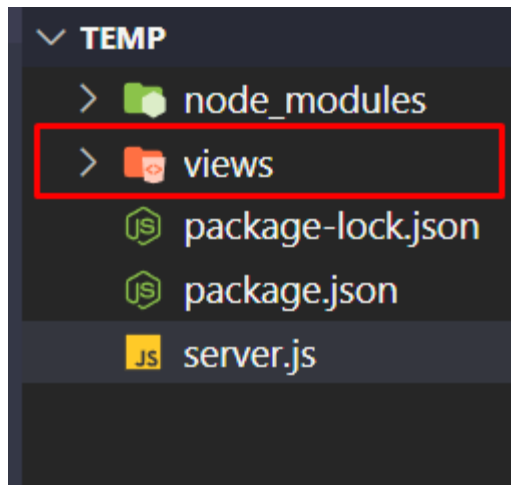
```
npm install -D nodemon
```

```
{  
  "name": "temp",  
  "version": "1.0.0",  
  "description": "",  
  "main": "server.js",  
   Debug  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1",  
    "start": "nodemon server.js"  
  },  
}
```

IMPORTANDO ARQUIVOS

Agora vamos importar nosso site para dentro do nosso servidor.

Crie uma nova pasta chamada de views e arraste seu conteúdo para dentro dela:



```
npm install nunjucks
```

Agora vamos instalar outra extensão para o nosso servidor o nunjucks

O nunjucks é uma template engine que é um pacote de funcionalidades novas para nosso server.

CHAMANDO O NUNJUCKS

```
const express = require ('express');  
const server = express ();  
const nunjucks = require ('nunjucks');  
  
server.set ("view engine","html")  
nunjucks.configure("views",{  
  express:server  
})
```

INICIANDO O SITE

```
server.get("/", function(req, res){  
  return res.render("index");  
});
```

-> dessa forma nos agora vamos renderizar nosso arquivo

HTML no caminho "/".

Teste

Como você pode perceber nosso arquivo css não respondeu, isso se por conta que não instanciamos ele ainda.

INICIANDO O CSS DO SITE

Primeiro vamos criar a linha de código `server.use(express.static('public'))`

Ele vai nos permitir utilizar arquivos estáticos que estejam dentro da pasta public.

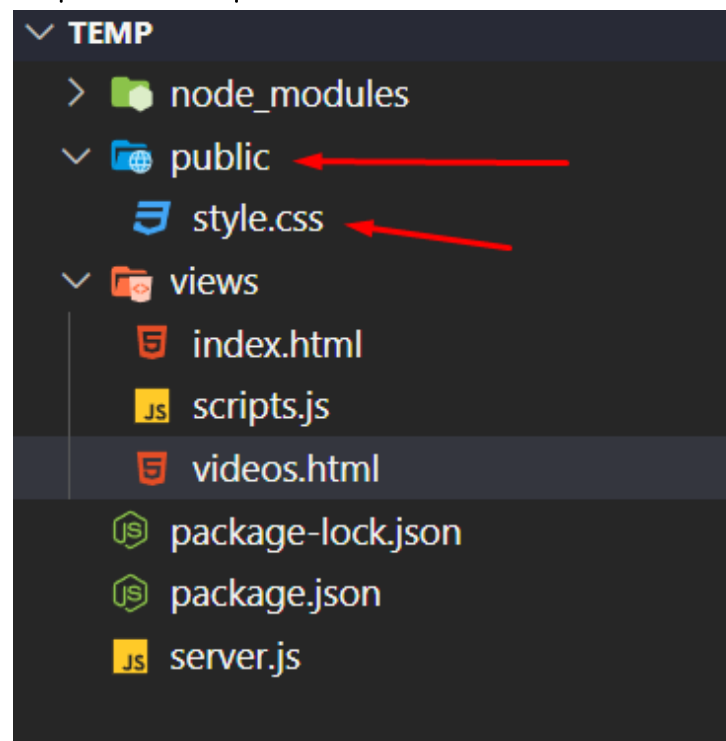
Que é uma pasta que vamos criar e arrastar nosso arquivo CSS pra dentro dela - Conceito public.

Vamos também criar a rota para nossa página de vídeos.

```
server.get("/videos", function(req, res){  
  return res.render("videos");  
})
```

INICIANDO O CSS DO SITE

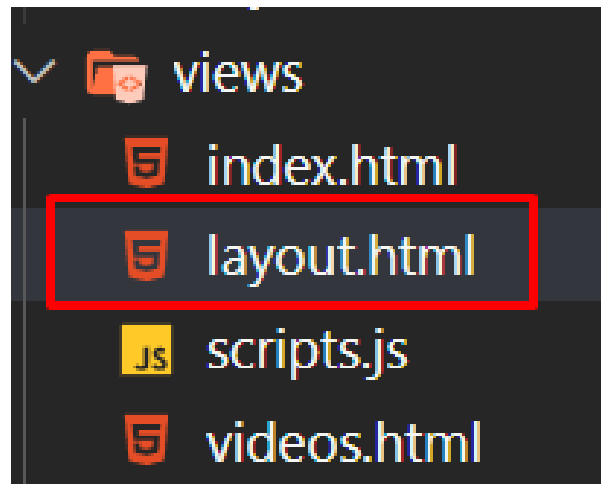
Crie a pasta public e coloque o arquivo css dentro dela..



APROVEITANDO DADOS

Primeiro crie um novo arquivo chamado layout.HTML.

depois copie todo os códigos do index.html para o arquivo layout.htm. Lembre-se de alterar o caminho no servidor.



```
server.js index.html videos.html layout.html package.json
views > layout.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <link rel="stylesheet" href="/style.css">
4
5 <head>
6
7 <title>Fernando Lucas</title>
8 </head>
9 <body>
10 <header>
11 <div class="links">
12 <a href="https://www.instagram.com/fernando.massan" target="_blank">Instagram</a>
13 <a href="https://www.linkedin.com/in/fernando-lucas-de-castro-309352233/" target="_blank">Linkedin</a>
14 <a href="https://www.github.com/namukuzedim" target="_blank">Github</a>
15 <a href="/videos">Videos</a>
16 </div>
17 </header>
18 <div id="box">
19 
20 <h1>Fernando Lucas <br> Macã - Minova</h1>
21 <h2>Programador Full-Stack</h2>
22 <p>Um dev que forma outros dev's, coordenador do <a href="https://estudeconosco.cevcolegio.com.br/sobre">C
23 </div>
24 <div id="linksFooter" class="links">
25 <a href="https://www.instagram.com/prof.fernandolucas" target="_blank">Instagram</a>
26 <a href="https://www.linkedin.com/in/fernando-lucas-de-castro-309352233/" target="_blank">Linkedin</a>
```

IMPORTÂNCIA DA INDENTAÇÃO DO CÓDIGO

Agora nós vamos utilizar códigos que já foram processados e renderizados, dessa forma vamos acelerar a velocidade com que conteúdos novos são mostrados.

agora você vai entender visualmente o motivo de
mantermos o código organizado e padronizado para
todas as páginas.

INICIANDO O REAPROVEITAMENTO

views > layout.html > html > body


```
1  <!DOCTYPE html>
2  <html lang="en">
3  <link rel="stylesheet" href="/style.css">
4
5  <head>
6
7      <title>Fernando Lucas</title>
8  </head>
9  <body>
10     <header>
11         <div class="links">
12             <a href="https://www.instagram.com/fernando.massan" target="_blank">Instagram</a>
13             <a href="https://www.linkedin.com/in/fernando-lucas-de-castro-309352233/" target="_blank">Linkedin</a>
14             <a href="https://www.github.com/namukuzedim" target="_blank">Github</a>
15             <a href="/videos">Videos</a>
16         </div>
17     </header>
18
19     {% block content %}
20
21
22     {%endblock%}
23
24 </body>
25 </html>
```

Va até o arquivo layout e apague todo o conteúdo que não se repete.

Depois adicione os comando {%%}.

INICIANDO O REAPROVEITAMENTO

Agora no arquivo

 *index.html*

Vamos o trocar o nome dele para about, e adicionar o comando `{%extends "lauout.html"%}`, o arquivo deve ficar assim.

```
{% extends "layout.html" %}
{% block content %}

    <div id="box">
        
        <h1>Fernando Lucas <br> Maçã - Minova</h1>
        <h2>Programador Full-Stack</h2>
        <p>Um dev que forma outros dev's, coordenador do <a href="https://estudeconosco.cevcolegio.com.br/sobre">
    </div>
    <div id="linksFooter" class="links">
        <a href="https://www.instagram.com/prof.fernandolucas" target="_blank">Instagram</a>
        <a href="https://www.linkedin.com/in/fernando-lucas-de-castro-309352233/" target="_blank">Linkedin</a>
        <a href="https://www.github.com/namukuzedim" target="_blank" >Github</a>
    </div>
</body>

{% endblock %}
```

Aproveitando dados

DESAFIO -

Use a lógica da funcionalidade `{%extends%}` na página de  `videos.html` ✕

```

views > videos.html > ...
1  {% extends "layout.html" %}
2
3  {% block title %}
4  <title>Fernando Lucas - videos dinamicos </title>
5  <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Ou
6
7  {% endblock %}
8
9
10 {% block content %}
11
12
13
14 > <section class="cards">...
53 </section>
54
55 > <div class="modal_overlay">...
67 </div>
68
69 </div>
70
71 <div id="linksFooter" class="links">
72 <a href="https://www.instagram.com/prof.fernandolucas" target="_blank">Instagram<
73 <a href="https://www.linkedin.com/in/fernando-lucas-de-castro-309352233/" target=
74 <a href="https://www.github.com/namukuzedim" target="_blank" >Github</a>
75
76 </div>
77
78 <script src="scripts.js"></script>
79 {%endblock%}

```

```

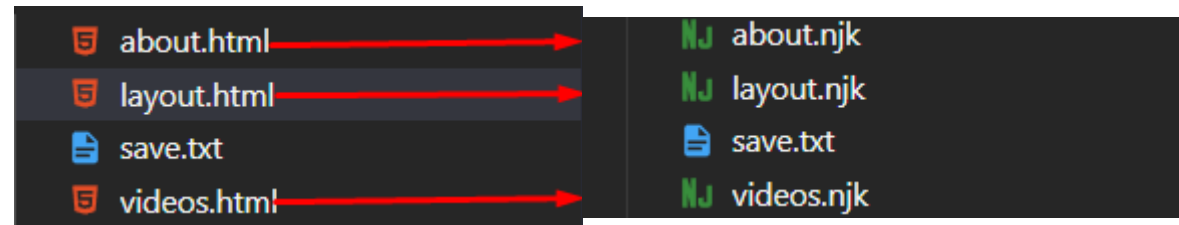
views > layout.html > html > title
1 <!DOCTYPE html>
2 <html lang="en">
3 <link rel="stylesheet" href="/style.css">
4
5  {% block title %}
6  <title>Fernando Lucas </title>
7  {% endblock %}
8
9 <head>
10
11 </head>
12 <body>
13 <header>
14 <div class="links">
15 <a href="https://www.instagram.com/fernando.massan" target="_blank">Instagram</a>
16 <a href="https://www.linkedin.com/in/fernando-lucas-de-castro-309352233/" target="_blank">Linkedin</a>
17 <a href="https://www.github.com/namukuzedim" target="_blank" >Github</a>
18 <a href="/videos">Videos</a>
19 </div>
20 </header>
21
22 {% block content %}
23
24
25
26 {%endblock%}
27
28
29 </body>
30 </html>

```

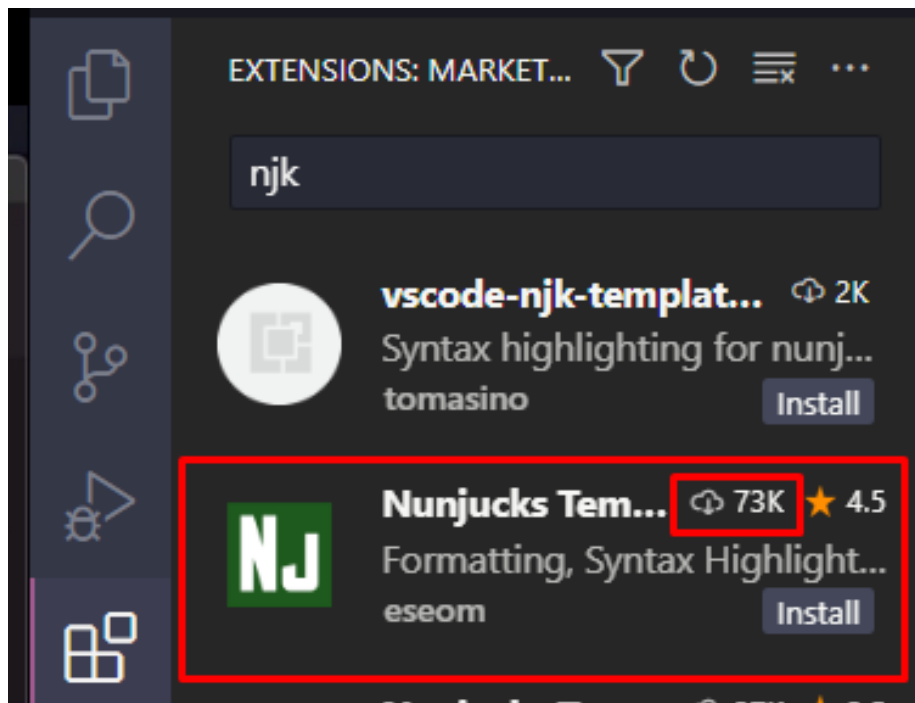
AGORA VAMOS SERVIR OS ARQUIVOS .NJK

Agora nós vamos transformar nossos arquivos HTML em NJK.

```
server.js
server.js > ...
1  const express = require('express');
2  const server = express();
3  const nunjucks = require('nunjucks');
4
5  server.use(express.static('public'))
6
7  server.set(["view engine", "njk"])
8  nunjucks.configure("views", {
9    express: server
10 })
11
12
13 server.get("/", function(req, res) {
14   return res.render("about");
15 })
16 server.get("/videos", function(req, res) {
17   return res.render("videos");
18 })
19
20 server.get("/")
21 server.listen(5000, function() {
22   console.log("server is running")
23 })
```



INSTALE UMA NOVA EXTENSÃO



AGORA REINICIE O VSCODE

VOLTANDO A COMPLETAR HTML

Você deve ter percebido que depois de instalar o a extensão para leitura do .njk, nossos arquivos html (njk) não estão se comportando como antes.

Isso acontece porque precisamos adicionar a configuração do HTML no Nunjucks.

VOLTANDO A COMPLETAR HTML

Nunjucks Template v0.5.1
eseom | 73,917 | ★★★★★ (4)
Formatting, Syntax Highlighting, Hover, and Snippets
Disable | Uninstall | Settings
This extension is enabled globally.

emmet.includeLanguages: {
 "njk": "html"
},

- For vscode embedded emmet, notify that **njk** is html file type

html.format.wrapLineLength: 120

- for vscode-icons ([issue #6](#))

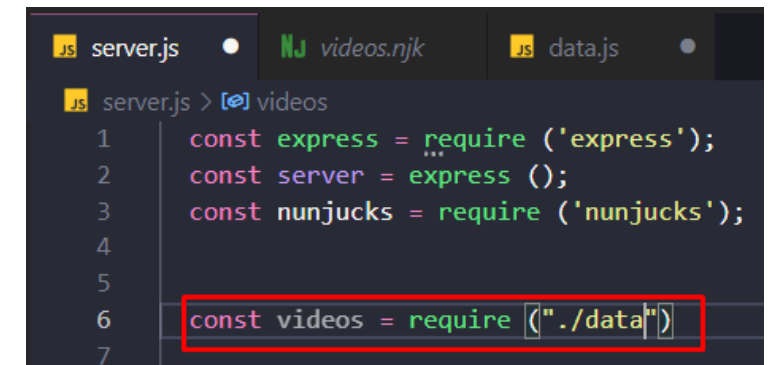
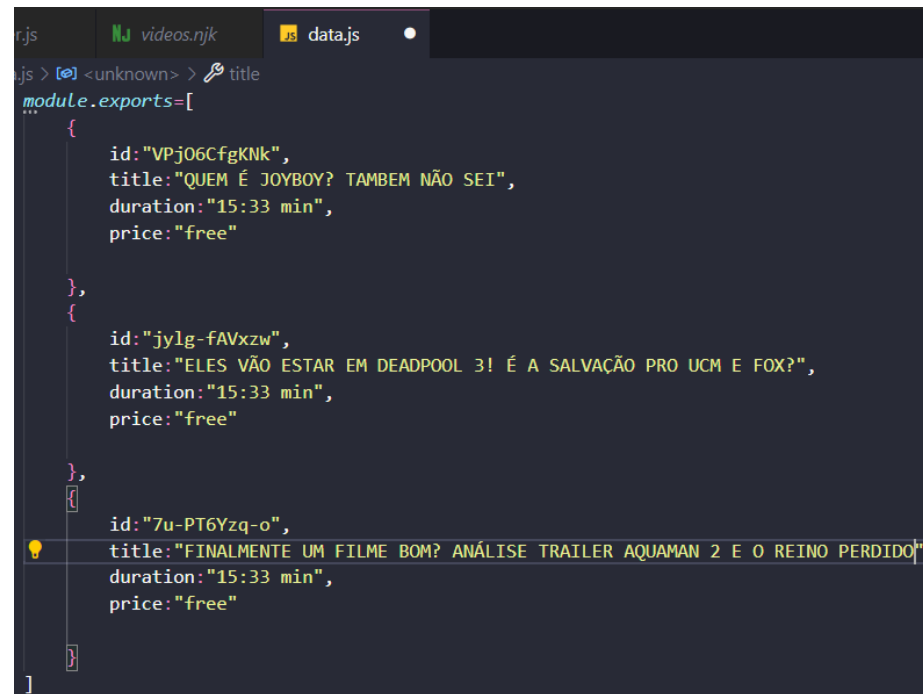
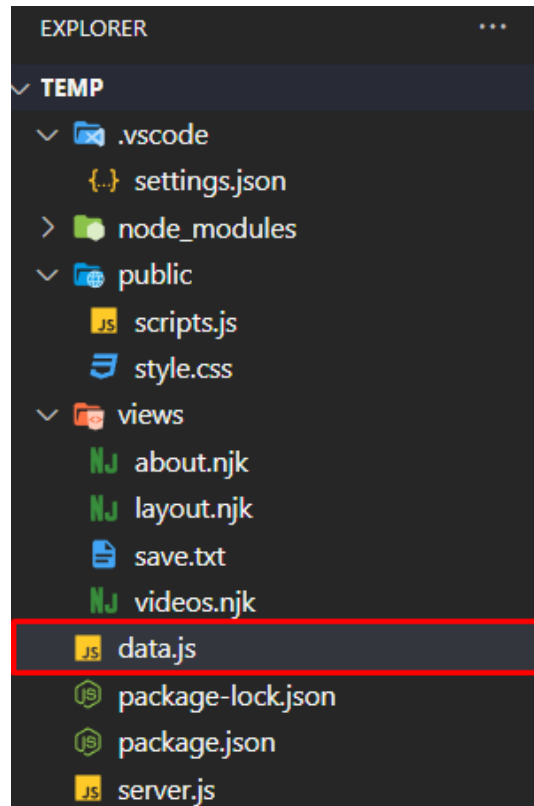
CTRL+SHIFT+P

>settings
Developer: Generate Color Theme From Current [Settings](#)

server.js settings.json × about.njk videos.njk

```
.vscode > settings.json > ...  
1 {  
2   "liveServer.settings.port": 5501,  
3   "emmet.includeLanguages": {  
4     "njk": "html"  
5   },  
6 }
```

EXPORTANDO E IMPORTANDO JAVASCRIPT



PASSANDO DADOS DO BACK PARA O FRONT

Apague todos os
cards. ->

```
server.js  videos.njk  data.js
server.js > server.get("/videos") callback > items
1  const express = require('express');
2  const server = express();
3  const nunjucks = require('nunjucks');
4
5  const videos = require('./data');
6
7
8
9  server.use(express.static('public'))
10
11  server.set('view engine', 'njk')
12  nunjucks.configure('views', {
13    express: server
14  })
15
16
17  server.get('/', function(req, res) {
18    return res.render('about');
19  })
20
21  server.get("/videos", function(req, res) {
22    return res.render("videos", {items: videos});
23  })
```

```
save.txt  videos.njk  data.js
videos.njk
endblock %}

block content %}

{% for item in items %}
  <section class="cards">
    <div class="card" id="{{item.id}}">
      <div class="card_image">
        
      </div>
      <div class="card_content">
        <p>{{item.title}} </p>
      </div>
      <div class="card_info">
        <p>{{item.duration}}</p>
        <p class="card_price"> {{item.price}} </p>
      </div>
    </div>
  {% endfor %}
</section>
```

DESAFIO -

Entenda como o css não está sendo respeitado no arquivo da página de vídeos e concerte.

AGORA VAMOS SUBSTITUIR O CONTEÚDO DA PÁGINA ABOUT PELO BACK-END.

```
<div id="box">|
  
  <h1>Fernando Lucas <br> Maçã - Minova</h1>
  <h2>Programador Full-Stack</h2>
  <p>Um dev que forma outros dev's, coordenador do <a href="https://estudeconosco.cevcolegio.com.br/sobre">C
</div>
<div id="linksFooter" class="links">
  <a href="https://www.instagram.com/prof.fernandolucas" target="_blank">Instagram</a>
  <a href="https://www.linkedin.com/in/fernando-lucas-de-castro-309352233/" target="_blank">Linkedin</a>
  <a href="https://www.github.com/namukuzedim" target="_blank" >Github</a>
</div>
</body>
```

NJ about.njk

JS server.js

```
server.get("/", function(req,res){
  const about = {
    avatar_url: "https://avatars.githubusercontent.com/u/117499108?v=4",
    name: "Fernando Lucas",
    role: "Programador Full-Stack",
    description: 'Um dev que forma outros dev',
    link: [
      {name: "Github", url: "https://github.com/namukuzedim"},
      {name: "Github", url: "https://github.com/namukuzedim"},
      {name: "Github", url: "https://github.com/namukuzedim"},
    ]
  }
  return res.render("about");
})
```

AGORA VAMOS SUBSTITUIR O CONTEÚDO DA PÁGINA ABOUT PELO BACK-END.

```
const about = {  
  avatar_url: "https://avatars.githubusercontent.com/u/117499108?v=4",  
  name: "Fernando Lucas",  
  role: "Programador Full-Stack",  
  description: 'Um dev que forma outros dev',  
  link: [  
    {name: "Github", url: "https://github.com/namukuzedim"},  
    {name: "Github", url: "https://github.com/namukuzedim"},  
    {name: "Github", url: "https://github.com/namukuzedim"},  
  ]  
}  
return res.render("about", {about});  
})
```

JS server.js

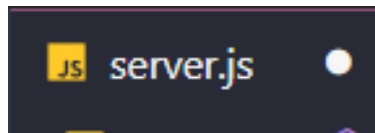
Precisamos também chamar nosso objeto dentro da renderização da página about, para que possamos usar ele dentro da página about.

AGORA VAMOS SUBSTITUIR O CONTEÚDO DA PÁGINA ABOUT PELO BACK-END.

```
server.set("view engine","njk")
nunjucks.configure("views",{
  express:server,
  autoescape:false
})
```

E para garantir que o nunjucks não vai segurar nenhum código HTML, vamos adicionar uma funcionalidade no server.

Autoscape:false



AGORA VAMOS SUBSTITUIR O CONTEÚDO DA PÁGINA ABOUT PELO BACK-END.

Agora é só substituir na nossa página about.

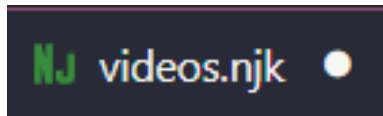
```
<div id="box">
  <img src= {{about.avatar_url}} alt="Deveria existir uma imagem aqui.">
  <h1> {{about.name}} <h1> <br> Maçã - Minova</h1>
  <h2>{{about.role}}</h2>
  <p>{{about.description}}</p>
</div>

<div id="linksFooter" class="links">

{% for link in about.link %}
  <a href="{{link.url}}" target="_blank">{{link.name}}</a>
{% endfor %}
```

NJ about.njk

VAMOS CRIAR VÍDEOS EM DESTAQUE.



Vamos utilizar nossa template engine para criar um if {%

```
<div class="card_image">
  Featured</div>
  {% endif %}
```

VAMOS CRIAR VÍDEOS EM DESTAQUE.

```
data.js > <unknown> > price
1  module.exports=[
2    {
3      id:"VPj06CfgKNk",
4      title:"QUEM É JOYBOY? TAMBEM NÃO SEI",
5      duration:"15:33 min",
6      price:"free",
7      featured: true
8    },
9    {
10     id:"jylg-fAVxzw",
11     title:"ELES VÃO ESTAR EM DEADPOOL 3! É A SALVAÇÃO PRO UCM E FOX?",
12     duration:"15:33 min",
13     price:"free"
14   },
15   {
16     id:"7u-PT6Yzq-o",
17     title:"FINALMENTE UM FILME BOM? ANÁLISE TRAILER AQUAMAN 2 E O REINO PERDIDO",
18     duration:"15:33 min",
19     price:"free",
20     featured: true
21   }
22 ]
```

No arquivo data vamos utilizar uma nova propriedade para os vídeos que vão ficar em destaque, que é a propriedade `feature:true`.

VAMOS CRIAR VÍDEOS EM DESTAQUE.

```
/* === featured === */  
.card_image{  
  position: relative;  
}  
  
.featured {  
  position: absolute;  
  background-color: #57a615;  
  color: #171515;  
  padding: 2px 8px;  
  right: 5px;  
  border-radius: 16px;  
  top: -6px;  
}
```

Agora no css vamos selecionar e fazer as alterações necessárias

VAMOS CRIAR VÍDEOS EM DESTAQUE.

Só para mantermos a organização e evitar futuros problemas vamos adicionar uma classe ao nosso css. Também vamos retirar todo tipo de cache.

```
<div class="card_image {{ 'isfeatured' if item.featured }}" >
  Featured</div>
  {% endif %}
```

```
server.set ("view engine","njk")
nunjucks.configure("views",{
  express:server,
  autoescape:false,
  noCache:true
})
```

```
/* === featured === */
.isfeatured.card_image{
  position: relative;
}

.featured {
  position: absolute;
  background-color: #57a615;
  color: #171515;
  padding: 2px 8px;
  right: 5px;
  border-radius: 16px;
  top: -6px;
}
```



Featured

QUEM É JOYBOY? TAMBEM
NÃO SEI

15:33 min

free



ELES VÃO ESTAR EM
DEADPOOL 3! É A SALVAÇÃO
PRO UCM E FOX?

15:33 min

free



Featured

FINALMENTE UM FILME BOM?
ANÁLISE TRAILER AQUAMAN
2 E O REINO PERDIDO

15:33 min

free

PÁGINA DE VÍDEO ÚNICO

Agora vamos criar uma página somente para nossos vídeos.
Vamos primeiro criar a rota no servidor para essa página.



```
server.get("/video",function(req,res){  
  const id = req.query.id  
  res.send(id)  
})  
  
server.listen(5000, function() {  
  (console.log ("server is running"))  
})
```

PÁGINA DE VÍDEO ÚNICO

Você de estar se perguntando o que é esse query.id


Você já deve ter navegado em algum site e ter visto essa URL.

```
localhost:5000/video?id=ldkaslçkdlçaksd
```

Nós vamos utilizar o id do nosso vídeo para carregar o vídeo em uma página só.

Chamamos essa estrutura de query.string

PÁGINA DE VÍDEO ÚNICO -

Vamos criar agora a função para rastrear dentro nosso arquivo  `data.js` os ids dos nosso vídeos.

```
server.get("/video",function(req,res){
  const id = req.query.id
  const video = videos.find(function(video){

    if (video.id==id) {
      return true
    }

  })

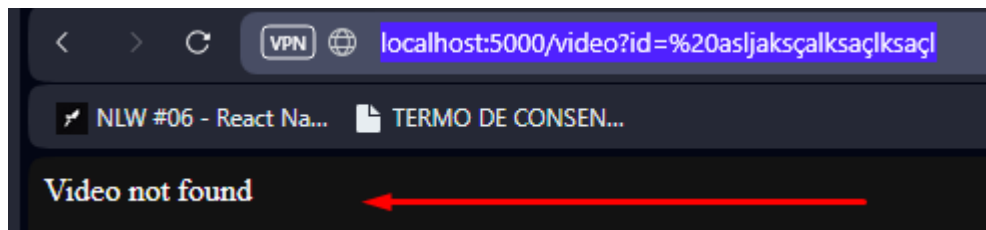
  if (!video){
    return res.send ("Video not found")
  }

  res.send(id)
})
```

PÁGINA DE VÍDEO ÚNICO -

Se tudo tiver dado certo, quando você tentar carregar o id aleatório.

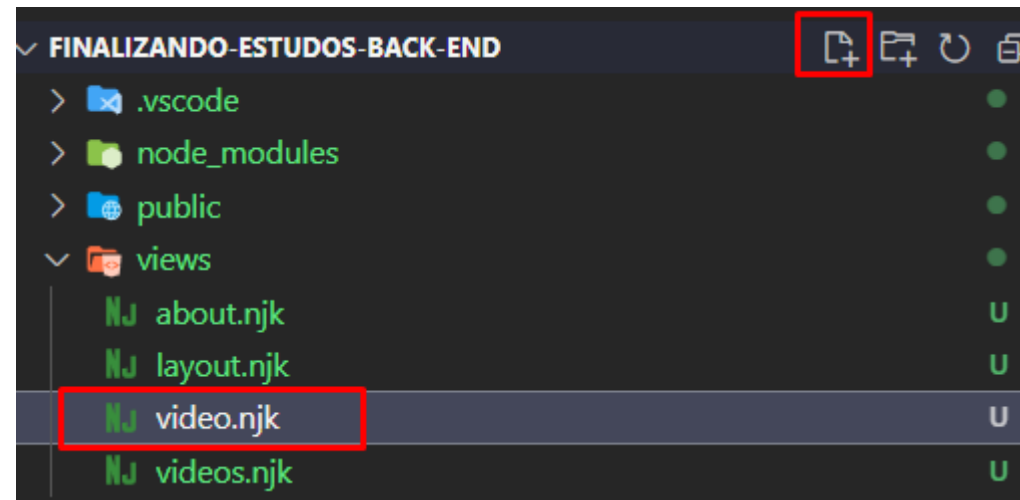
Vai aparecer a seguinte mensagem.



PÁGINA DE VÍDEO ÚNICO -

Agora vamos terminar de criar a rota para nossa página de vídeo único.

```
server.get("/video",function(req,res){  
  const id = req.query.id  
  const video = videos.find(function(video){  
  
    if (video.id==id) {  
      return true  
    }  
  
  })  
  
  if (!video){  
    return res.send ("Video not found")  
  }  
  
  return res.render ("video",{item: video})  
})
```



PÁGINA DE VÍDEO ÚNICO -

Vamos configurar nosso novo arquivo vídeo.njk.

Copie todo o conteúdo de vídeos.njk para vídeo.njk.

Fazendo pequenas alterações e retirando algumas Tag's que não vamos utilizar.

```
video.njk
{% extends "layout.njk" %}

{% block title %}
<title>Fernando Lucas - videos dinamicos </title>
<link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@24,400,1,0" />
{% endblock %}

{% block content %}

<section class="cards">
{% for item in items %}
<div class="card" id="{{item.id}}">
  <div class="card_image {{ 'isfeatured' if item.featured }}">
    Featured</div>
    {% endif %}
  </div>
  <div class="card_content">
    <p>{{item.title}} </p>
  </div>
  <div class="card_info">
    <p>{{item.duration}}</p>
    <p class="card_price">{{item.price}} </p>
  </div>
</div>
</div>
```

PÁGINA DE VÍDEO ÚNICO -

Para testar se tudo está ok. Vamos utilizar uma id real dos nossos vídeos.

```
localhost:5000/video?id=jylg-fAVxzw
```

Assim podemos carregar nossa página de vídeo.

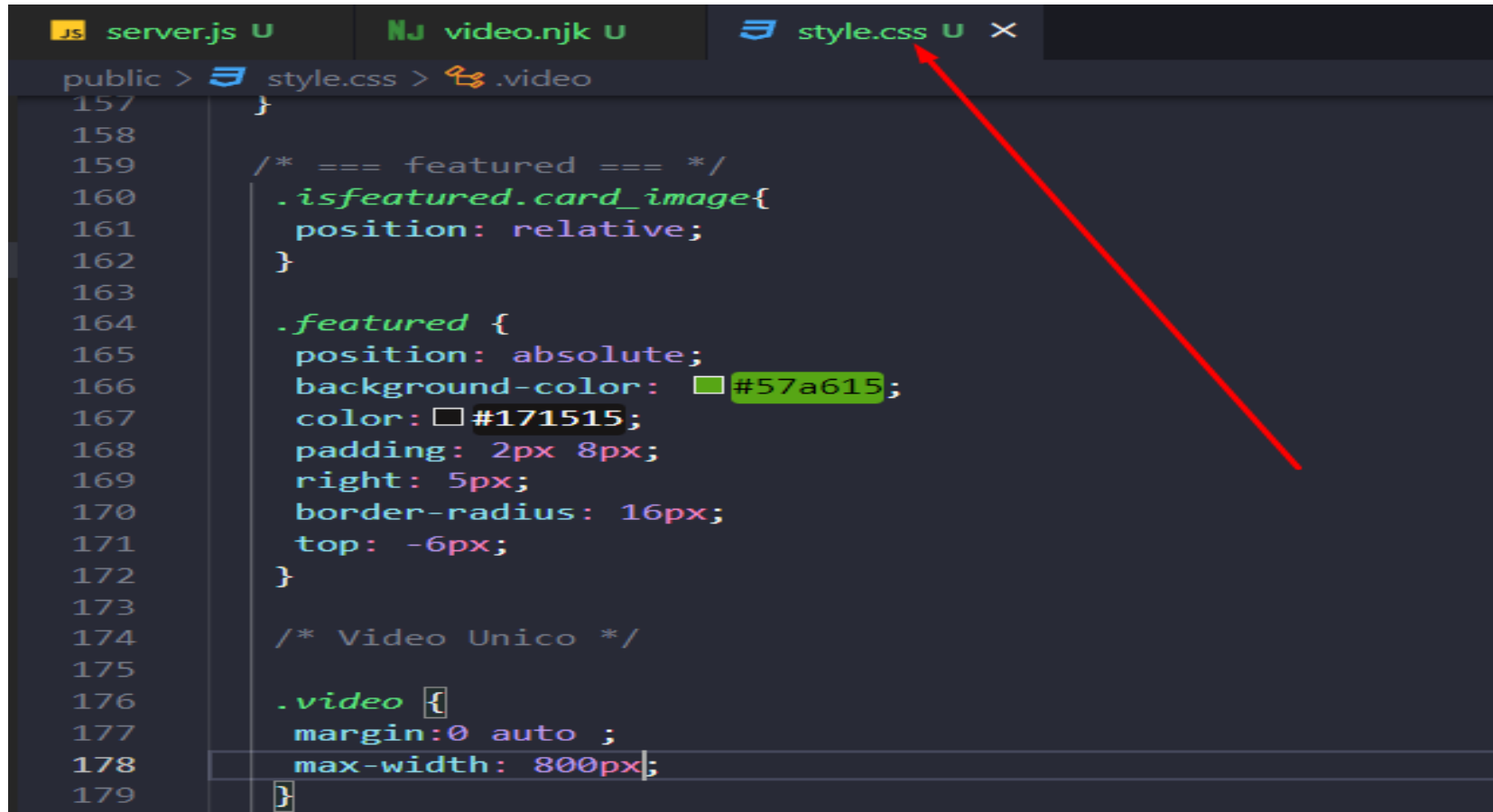
No nosso arquivo `video.njk` vamos estilizar novamente nosso vídeo.

PÁGINA DE VÍDEO ÚNICO -

```
{% block content %}
    <section class='video">
        <div class="card" id="{{item.id}}">
            <div class="card_image {{ 'isfeatured' if item.featured }}" >
                Featured</div>
                {% endif %}
            </div>
            <div class="card_content">
                <p>{{item.title}} </p>
            </div>
            <div class="card_info">
                <p>{{item.duration}}</p>
                <p class="card_price"> {{item.price}} </p>
            </div>
        </div>
    </div>
</section>

{%endblock%}
```

PÁGINA DE VÍDEO ÚNICO -



```
server.js U  video.njk U  style.css U X
public > style.css > .video
157 }
158
159 /* === featured === */
160 .isfeatured.card_image{
161   position: relative;
162 }
163
164 .featured {
165   position: absolute;
166   background-color: #57a615;
167   color: #171515;
168   padding: 2px 8px;
169   right: 5px;
170   border-radius: 16px;
171   top: -6px;
172 }
173
174 /* Video Unico */
175
176 .video {
177   margin: 0 auto ;
178   max-width: 800px;
179 }
```

PÁGINA DE VÍDEO ÚNICO -

Mais alterações na página de vídeos.njk

```
{% block content %}
    <section class="video">
        <div class="card_only" id="{{item.id}}">
            <iframe src="https://www.youtube.com/embed/{{item.id}}" frameborder="0"></iframe>
        </div>
        <div class="card_content">
            <p>{{item.title}} </p>
        </div>
        <div class="card_info">
            <p>{{item.duration}}</p>
            <p class="card_price"> {{item.price}} </p>
        </div>
    </div>
</div>

</section>

{%endblock%}
```


PÁGINA DE VÍDEO ÚNICO -

E mais alterações na página de css.

```
/* Video Unico */

.video {
  margin: 0 auto ;
  max-width: 800px;
}

.card_only {
  position: relative;
  padding-top: 62.5%;
}

.card_only iframe {
  width: 100%;
  height: 90%;
  position: absolute;
  top: 5%;
}
```

PÁGINA DE VÍDEO ÚNICO -

Agora vamos esquecer um pouco a ideia do modal e apontar o usuário da aplicação direto para a página vídeo único quando ele clicar no card.

```
const modal_overlay = document.querySelector ('.modal_overlay');  
const cards = document.querySelectorAll ('.card');  
  
for (let card of cards ) {  
  card.addEventListener ("click",function(){  
    const videoId = card.getAttribute ("id");  
  
    window.location.href = `/video?id=${videoId}`  
  })  
};
```